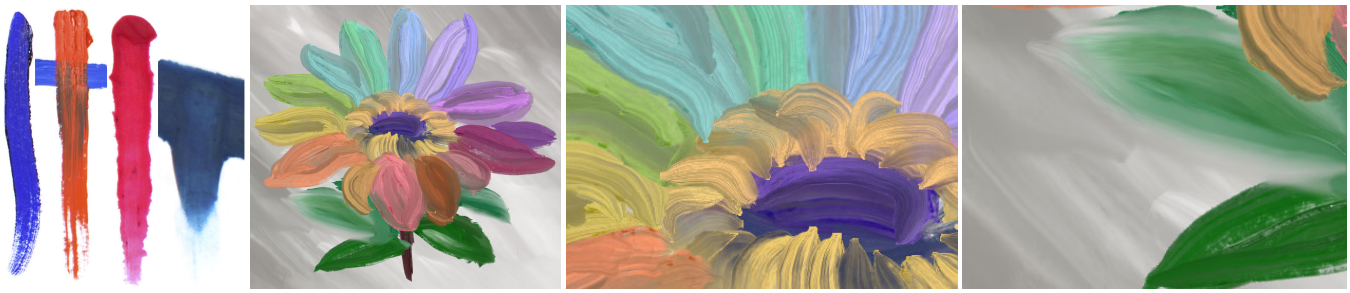
RealBrush: Painting with Examples of Physical Media

Jingwan Lu[1] Connelly Barnes[2] Stephen DiVerdi[2,3] Adam Finkelstein[1]
[1]Princeton University [2]Adobe Systems Inc. [3]Google Inc.



(a) (b) (c) (d)

Figure 1: *A simple painting created by our system. Left to right: (a) shows oil (left) and plasticine (right) exemplars which are used to synthesize the painting in (b). The foreground flower strokes use oil exemplars, while the background strokes use plasticine and are smoothed with our smudging tool using. (c) (d) show close-ups of the smearing and smudging effects.*

Abstract

Conventional digital painting systems rely on procedural rules and physical simulation to render paint strokes. We present an interactive, data-driven painting system that uses scanned images of real natural media to synthesize both new strokes and complex stroke interactions, obviating the need for physical simulation. First, users capture images of real media, including examples of isolated strokes, pairs of overlapping strokes, and smudged strokes. Online, the user inputs a new stroke path, and our system synthesizes its 2D texture appearance with optional smearing or smudging when strokes overlap. We demonstrate high-fidelity paintings that closely resemble the captured media style, and also quantitatively evaluate our synthesis quality via user studies.

CR Categories: I.3.4 [Computer Graphics]: Graphics Utilities

Keywords: stroke, stylization, data-driven, example, painting

Links: DL PDF WEB

1 Introduction

Traditional artists working with natural media take advantage of a great abundance of different materials. These can have different chemical properties, such as wet, dry, or cracked paint. Pigments can be scratched, smeared, or mixed with binders or thinners. Materials can have 3D relief or embedded particles. The artist can express her creativity by utilizing materials found in the wild, limited only by her imagination. In contrast, digital artists are heavily restricted. High quality approximations of commonly used media such as oil and watercolor have been achieved in research systems [Baxter et al. 2004; Chu and Tai 2005]. However, these systems rely on complex simulations that cannot easily generalize to new media. Commercial tools such as Adobe Photoshop achieve a wider range of effects with procedural algorithms, at the cost of requiring significantly more effort to generate a similar level of natural media fidelity. Ultimately, artists are limited by the built-in assumptions embodied by digital painting software.

We introduce "RealBrush," a system that allows artists to paint digitally with the expressive qualities of any physical medium. As input to the system, the artist first makes some example strokes demonstrating the medium's behaviors and scans them in. The scanned images are processed to make a "library." Within RealBrush, the user can select this library and make new strokes that are synthesized using the library to create novel marks in the same style. In this manner, the artist can digitally paint with whatever physical media she feels most comfortable, without requiring the development of a custom simulation algorithm (see Figure 2). We accomplish this by using a data-driven approach that derives the full knowledge of a medium from the example library, making only the minimal necessary assumptions about physicality and locality.

Expressive media can achieve different appearances based on the artist's skill and technique. In constructing an oil paint library, the artist may thickly apply paint, or may make smooth, textureless strokes. A watercolor artist may paint strokes in a calligraphic style. Therefore, libraries encode not only the physical properties of the media, but also their application in a specific style by a trained artist. With a calligraphic watercolor library, a novice user may make strokes of much higher quality than he could with a real brush.

The shape of individual strokes is only a small part of the behavior of physical media—traditional painting is not possible without the complex interaction between strokes. Wet pigment can be advected by subsequent strokes to create smears, or particles may be pushed around the canvas by finger smudges. The myriad of potential effects creates an undue burden to attempt to capture examples of all different behaviors for a library. A generic data-driven algorithm that is not tailored for natural media could easily become impractical due to requiring intractably large amounts of data.

Our main contribution is the plausible reproduction of natural media painting, with its many behaviors, tools, and techniques, in a practical and fully data-driven system. This is possible because we factorize the range of physical behaviors into a tractable set of orthogonal components that can be treated independently, which is motivated by our analysis of the space of natural media. We discuss the implications of this factorization, and present algorithms for capturing and reproducing each of these behaviors. Our results

ACM Reference Format
Lu, J., Barnes, C., DiVerdi, S., Finkelstein, A. 2013. RealBrush: Painting with Examples of Physical Media. ACM Trans. Graph. 32, 4, Article 117 (July 2013), 12 pages. DOI = 10.1145/2461912.2461998 http://doi.acm.org/10.1145/2461912.2461998.

DOI: http://doi.acm.org/10.1145/2461912.2461998

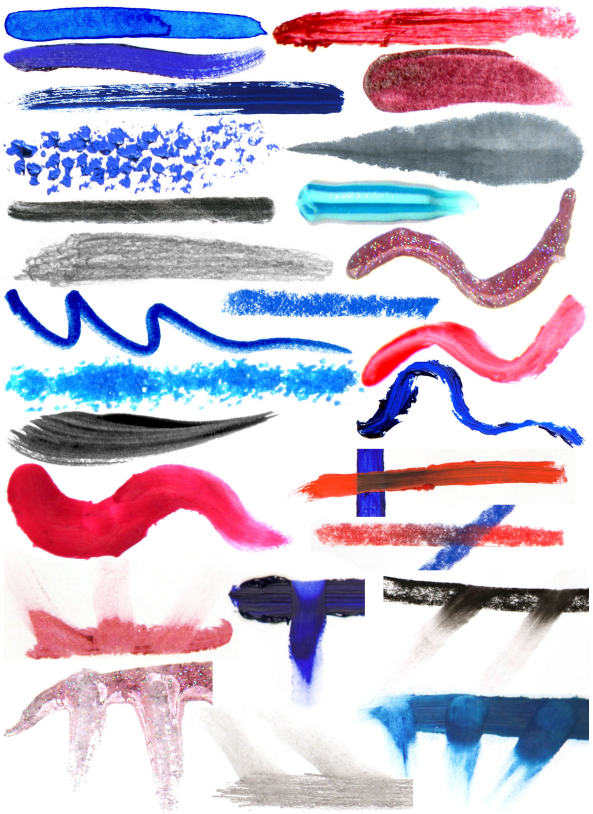


Figure 2: *Acquired natural media samples, including paint, charcoal, pastel, marker, lip gloss, plasticine, toothpaste, and glitter.*

demonstrate the algorithm with images painted by artists using a variety of different captured media and with a quantitative evaluation of the realism achieved by RealBrush.

2 Related Work

Despite the remarkable progress made by digital painting software to perform increasingly sophisticated image processing, it remains difficult to achieve results that mimic the textural qualities of real natural media. Established results fall into three broad categories.

Procedural approaches rely on heuristics explicitly designed to mimic specific natural media behaviors and effects, based on the developer's intuition [DiVerdi et al. 2012]. Commercial packages such as Adobe Photoshop use these techniques. They provide a wide range of effects for digital artists, but they have difficulty reproducing the visual fidelity of natural media and can require significant user effort to achieve a convincing level of realism.

For higher fidelity results, researchers have developed *simulation approaches* that numerically model the physical interaction between the virtual brush, the canvas, and the pigment medium. Specifically, modeling deformable 3D brushes and simulating paint fluid flow through a canvas are increasingly common, and have been applied successfully for particular natural media, including watercolor [Chu and Tai 2005], oil paint [Baxter et al. 2004; DiVerdi et al. 2010], and pastels [Van Haevre et al. 2007], with impressive results. Commercially, Microsoft's Fresh Paint application implements recent work [Chu et al. 2010; Baxter and Govindaraju 2010] to create an oil paint system. However, these results are all carefully tailored to their respective natural media, and extending any of them to other types of artistic tools is difficult. Our focus is a more general digital painting system that can support arbitrary natural media as supplied by the artist while maintaining high visual fidelity.

Figure 3: *Single stroke libraries for oil paint and plasticine.*

Our work belongs to the class of *data-driven approaches*, which includes such work as modeling virtual brushes [Baxter and Govindaraju 2010; Xu et al. 2004], generating brush stroke paths [Xu and Dong 2006; Xie et al. 2012], and simulating pigment effects [Xu et al. 2007]. Perhaps most straightforward is using scanned marks as basic drawing units (footprints), interpolated along a path to produce novel strokes [Xie et al. 2011]. However these techniques all still depend on physical simulations and procedural rules which limit the fidelity and generality of their results. For realistic synthesis of arbitrary media, we use a solely data-driven approach and avoid any physical model or procedural rules.

Researchers have deformed strokes to match new paths. Zeng et al.'s work on applying painterly styles to photographs [2009], and Xu et al.'s decomposition and deformation of Chinese paintings [2006] both warp images of real strokes to create novel shapes. Earlier work by Hsu et al. [1994] developed a system based on stylizing lines with warped pieces of art, which has become a core algorithm of commercial programs such as Adobe Illustrator. All of these systems suffer from a common limitation though, that they can only support a small range of deformations of example strokes before the altered appearance becomes unrealistic. Zhou et al. [2013] achieve a wider range of deformations, but only from individual exemplars with uniform texture.

Most similar to our approach is the work of Ando and Tsuruno [2010], and Kim and Shin [2010], which use images of real strokes to stylize user input lines. These works focus on single stroke synthesis, which is a part of our approach, but we additionally explore data-driven smearing and smudging. Both approaches make aggressive simplifications, losing fidelity to gain performance. That each system relies on small patches that are not deformed fundamentally limits these techniques by forcing many patch boundaries within a stroke, which is frequently where artifacts occur. The lack of deformation also reduces the range of shapes they can synthesize from few example strokes, and therefore increases the computational burden per stroke by requiring more search. These small patches further limit the scale of texture features that can be reliably reproduced from example media, because patch boundaries can cause jarring discontinuities. RealBrush has two significant advantages over these approaches. First, we are able to support a wider array of artistic media. Second, we gain higher fidelity: we are able to reproduce strokes that are indistinguishable from real examples, as demonstrated by our user study.

Also related is work in texture synthesis. Wang et al. [2004] generate textures sampled from paintings to stylize 2D images. Yan et al. [2008] apply a similar technique for shading 3D geometry. Neither of these approaches provide a means for interactive, paint-style control. Conversely, Schretter [2005] and Ritter et al. [2006] both demonstrate brush-based painting systems that use texture synthesis to fill arbitrary regions with acquired samples. However, they do not support oriented anisotropic textures, and they do not consider interaction between overlapping textured regions.

Preprocessing (§4)

Single Stroke Synthesis:

- Find optimal sequence of library samples (§5)

choice:
- Alpha blending (§5.1)
- Graph cut (§5.1)
- Texture synthesis (§5.2)

Stroke Interaction Synthesis:

- Detect overlap regions in query stroke (§6.1)
- Find closest neighbor example (§6.2)
- Vectorize interaction regions (§6.3)

choice:
- Apply smear operator (§6.5)
- Apply smudge operator (§6.6)

Figure 4: *Algorithm overview.*

3 Understanding Natural Media

Natural media exhibit a tremendous number of different types of behaviors (see Figure 2). Simulation based approaches rely on complex physical models to control these effects from a small number of parameters. They pose a daunting problem for data-driven approaches, because of the huge number of different examples that need to be acquired to reasonably approximate the medium.

Factorization. We factor the effect space into four orthogonal bases—"shape," "smear," "smudge," and "composite"—that make the data acquisition and search problem tractable. Shape refers to the silhouette and texture of a single stroke made in isolation. Many obvious media features are part of shape including watercolor's darkened edges. However, most of the important qualities of natural media are due to the interactions among multiple strokes. Smear is exemplified by applying a new stroke across wet paint, "dirtying" the brush and smearing the two paint colors together (Figure 6b,c). Smearing is the core component of color blending on canvas, and is also referred to as "bidirectional" pigment transfer (from the brush to the canvas, and vice versa). Smudge specifically refers to stroking over wet paint with a clean brush or finger to make a smudge mark, but we use it more generally to refer to any action that modifies pigment on canvas without applying more pigment, including using other implements such as salt, a blowdryer, or tissue paper (Figure 6d,e). Finally, composite refers to the way colors mixed or applied atop one another optically combine to form the final reflected color, such as in glazing or overlapping transparent strokes, or during bidirectional pigment transfer.

Each basis can be captured in isolation, reducing the burden for data-driven acquisition and synthesis. Shape exemplars are isolated strokes. Smear exemplars come from crossing paint strokes of two different colors, so the mixing proportion can be determined by color. Smudge exemplars do not have a foreground color, so registered images of the canvas paint are needed, before and after the smudge is applied. Composite exemplars are overlapping strokes of different colors, so the combined color can be sampled.

In proposing this factorization, we reduce the library size that must be acquired from n^4 to $4n$, where n is the number of examples needed for each basis. We leave composite as future work, and focus on achieving realistic results for shape, smear, and smudge. See Figure 2 and Section 4 for examples of these effects.

Algorithm Assumptions. Procedural and simulation based approaches to digital painting encode explicit assumptions about the character of natural media into their algorithms. Simulations assume properties of the underlying physical system, while procedural rules assume specific behaviors the media exhibits. Data-driven techniques have the ability to ease these assumptions, but completely removing them may not be possible. We enumerate the basic assumptions we rely on in data acquisition and synthesis.

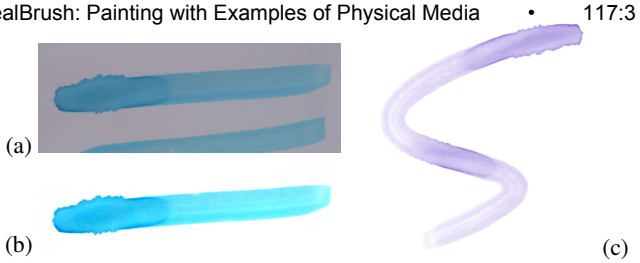


Figure 5: *Workflow overview. (a) Paint strokes are acquired with a handheld, point-and-shoot camera with indoor lighting. (b) A few simple edits isolate and white balance the stroke. (c) The library stroke is piecewise matched and deformed to the input curve.*

We call our first assumption *stroke causality*—that pixels that have been touched will not be modified after the stroke has passed. This means we explicitly cannot support effects like watercolor backruns, where excess water advects pigment into areas of the stroke that have already been painted.

The second assumption is *effect locality*—that a stroke's effect does not extend beyond its silhouette. This means that for effects like feathered watercolor where the pigment has advected away from the brush-canvas contact area, the brush silhouette must be made artificially large to encompass the entire final extent of the stroke.

Our final assumption is *recursion*. Given n strokes on a canvas, the $n+1$ stroke may have interactions that depend on the full, ordered set of previous strokes. Instead, we treat each stroke as interacting with a single, flattened canvas raster, which elides details about occlusions and ordering. This assumption enables efficient stroke interaction synthesis for paintings of thousands of strokes or more.

Media Texture. One complicating aspect of natural media, even in light of our factorization, is the wide variety of types of texture and appearance that are exhibited within each of our bases. Consider the following media and the characteristics of their shapes: watercolor has smooth internal textures with some granularity and sharp silhouettes. Oil paint has directional texture and sparse large features (blobs). Oil sponge has rough and sparse texture with sharp boundaries. Toothpaste has strong 3D structure. Glitter has noisy texture and sporadic application. Lip gloss has 3D structure and sparse noise texture. This is just a selection of possible variations.

Any single technique that could reliably synthesize all of these media efficiently would be a major advance in texture synthesis. Therefore it is clear that a small toolbox of core algorithms will be necessary to produce realistic results across all media.

Algorithm Overview. Our factorization of natural media behaviors suggests a factored algorithm. RealBrush is a data-driven painting implementation of this theory including shape, smear, and smudge behaviors, which are each treated separately and then combined.

First, shape, smear, and smudge examples are collected and processed to generate media libraries (Section 4). During painting, each stroke input by the user is processed individually. Its shape and texture are synthesized in isolation from a single stroke library (Section 5). Then, if the stroke covers paint already on the canvas, a smear effect is synthesized from a smear library and applied to the stroke (Section 6). When smudging, no new stroke is being generated, so a smudge effect is synthesized from a smudge library and directly applied to the paint on the canvas instead.

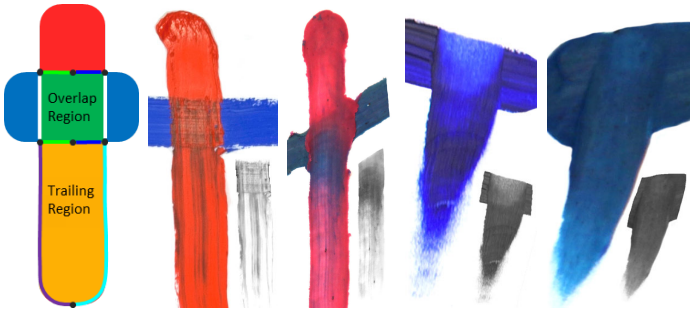
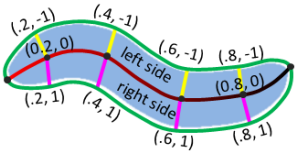


Figure 6: *Smearing and smudging exemplars. (a) We detect the overlap and the trailing regions. They are highlighted in green and yellow and are represented by eight polyline segments; (b) (c) show oil and plasticine smearing exemplars; (d) (e) show oil and plasticine smudging exemplars. The extracted smearing and smudging operators are inset.*

4 Example Collection and Processing

We collect libraries of exemplars made by real physical media (Figures 2 and 3) and provide them as default options for casual users. Advanced users can acquire their own libraries in the same manner. In light of our natural media factorization, we collect separate, different examples for each of shape, smear, and smudge. Our exemplars were captured using a DSLR camera mounted on a tripod under normal in-door lighting without any camera calibration. However, capture a handheld point-and-shoot camera also yields good results (Figure 5). We scan the exemplars into raster textures, and semi-automatically process them to create libraries. Single stroke (shape) exemplars are discussed in Section 4.1, while smear and smudge exemplars are in Section 4.2.

4.1 Single Stroke Processing



For each medium, we collect isolated brush strokes of different shape, curvature and thickness to build a single stroke library, $\mathbb{L}$, typically between 20 and 30 strokes. For each stroke, the user specifies a rough spine from start to end (inset, red to black line), which is smoothed and discretized into samples spaced a few pixels apart (inset, six black dots along spine). The stroke outline (green) is automatically extracted plus a small margin to preserve boundary effects. Ribs connecting the spine to the outline are added greedily, avoiding intersections (inset, yellow/magenta lines). Each stroke $\mathbf{L} \in \mathbb{L}$ consists of between 30 and 100 samples, $\mathbf{L} = \{\mathbf{t}\}$, where a sample $\mathbf{t} = \{\mathbf{x}, \mathbf{l}, \mathbf{r}\}$ consists of the 2D positions of the spine, left outline, and right outline points respectively. The stroke is parameterized by $u \in [0, 1]$ along the spine and $v \in [-1, 1]$ from left to right along each rib. We use the Multilevel B-splines Approximation (MBA) library [Hjelle 2001] to interpolate u, v coordinates for every pixel in the stroke. The output of this stage consists of the gray-scale intensity image of the exemplar, the vector representation, and the uv parameterization.

4.2 Overlapping Stroke Processing

Smearing is the result of transferring pigment from the canvas to the brush during a stroke, causing a mixing of colors (see Figure 6). Different physical media properties cause unique smearing behaviors. For example, thick wet oil paint smears significantly, pastels smear somewhat, and dry watercolors do not smear at all. Unconventional media such as plasticine smear differently without mixing pigments but still smudging them along. We use the term smearing for all stroke interactions containing overlapping strokes of different colors.

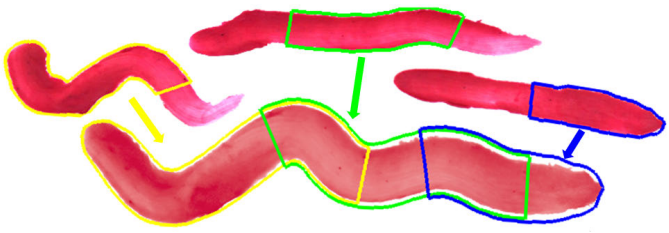


Figure 7: *Single stroke piecewise matching and warping. The bigger query stroke at the bottom is broken into three overlapping segments each of which matches an exemplar segment. The three segments are highlighted in yellow, green and blue.*

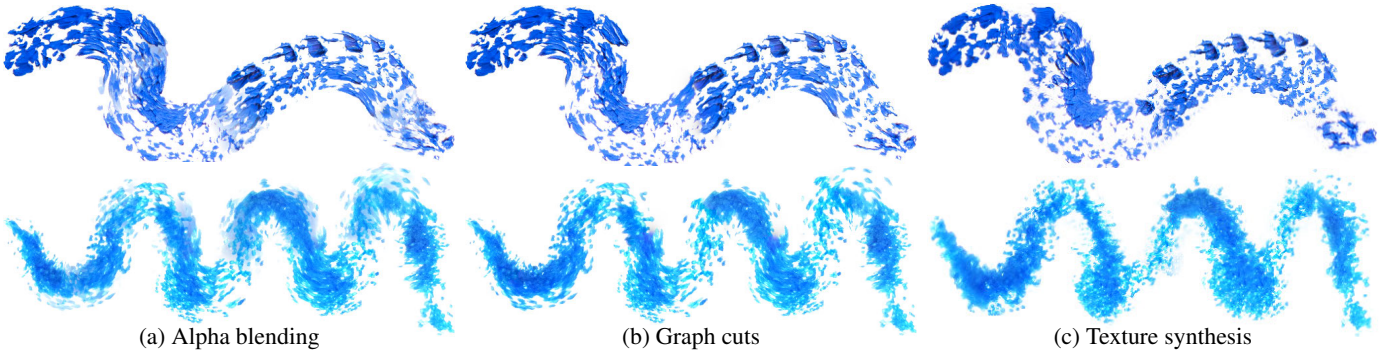
Artists can also smudge strokes on the canvas without applying additional pigment, commonly with a finger or clean brush. Such smudging exemplars contain two strokes of the same color—a background, and a smudged foreground.

For smearing and smudging separately, we collect between 10 and 20 pairs of overlapping strokes in several different media. For smearing, we collected oil, plasticine, and pastel. For smudging, we collected oil, plasticine, lipstick, charcoal, and pencil. Each pair of strokes overlap at different crossing angles and with different relative thickness. We only collect pairs of strokes interacting—our synthesis algorithm can plausibly reproduce the interactions among many strokes from this data. To process the scanned overlapping strokes, the user needs to specify the stroke spines (in our system) and create binary foreground and background masks, $\mathbf{F}$ and $\mathbf{B}$ (with Photoshop threshold and brush tool). The system then automatically vectorizes the foreground and background strokes (Section 4.1), and then extracts information specific to smearing and smudging. Note that though the exemplars shown here are near-orthogonal crossings, our synthesis algorithm can handle all possible overlap cases.

Interaction Region Vectorization: As input for the on-line synthesis of smearing or smudging, we extract binary masks for the overlap region $\mathbf{\Omega}$ and the trailing region $\mathbf{\Psi}$ (green and yellow regions in Figure 6a) by doing logic operations on the foreground and the background stroke masks. Specifically, $\mathbf{\Omega} = \mathbf{F} \cap \mathbf{B}$, where $\mathbf{F}$ and $\mathbf{B}$ are the foreground and background stroke masks. Let $\mathbf{u}(\mathbf{\Omega})$ define the average u coordinate of a connected region, then $\mathbf{\Psi} = \{\mathbf{A} \mid \mathbf{A} \subseteq \mathbf{F} \setminus \mathbf{\Omega}, \mathbf{u}(\mathbf{A}) > \mathbf{u}(\mathbf{\Omega})\}$ We refer to interaction region $\mathbf{\Pi} = \mathbf{\Omega} \cup \mathbf{\Psi}$ as the union of the overlap and the trailing regions. We then extract the contours of the interaction region as eight polylines (colored line segments in Figure 6a) which are used for synthesis (Section 6.3).

Smearing Operator: In the interaction region, the background blue pigments get mixed with the foreground red pigments resulting in vertical streaks of brownish color along the stroking direction (Figure 6b,c). Since the physical smearing process is more or less independent of the specific colors being mixed, it is safe to require the foreground to be red and background to be blue and obtain the mixing behaviors that are universal for paints of any colors. For each exemplar, we extract a *smearing operator* that is a 1 channel, 2D texture indicating the amount of background paint (from 0 to 100 percent) at each pixel. Since blue and red are more or less orthogonal after capturing, we simply use one minus the blue channel (lower right corner of Figure 6b,c). At runtime, we use the smearing operator to synthesize the smearing appearance of arbitrary colors (Section 6.5).

Smudging Operator: The foreground smudged strokes consist of advected pigment from the background strokes (Figure 6d,e). We



(a) Alpha blending (b) Graph cuts (c) Texture synthesis

Figure 8: *Artifacts in synthesis and methods to fix them. While simple alpha blending works for most media, some media may not be able to find well-registered matches and produce ghosting artifacts (a). Graph cuts often improves this (b). Texture distortion artifacts can also be present for highly textured media. These artifacts can be improved with our off-line texture synthesis module (c).*

extract a *smudging operator* (the lower right corners of Figure 6d,e) as the inverse intensity indicating the amount of background pigment at every pixel. In addition, we also extract a blending attenuation mask automatically by blurring the left, right, and upper side of the overlap region binary mask $\mathbf{\Omega}$. For better synthesis quality, a more precise blending attenuation mask can be provided by the user using standard image editing tools. The blending attenuation mask is used in the synthesis to provide gradual transition at the overlap region boundaries (Section 6.6).

Output. After processing is complete, the following information is passed to the online synthesis stage. Each of the smearing and smudging exemplars consist of the vectorization of both the foreground and background strokes and the vector representation of the interaction region. In addition, the smearing exemplars contain the smearing operator. The smudging exemplar contains the smudging operator and the blending attenuation mask. Optionally, the user can also provide as input the background strokes themselves before being smeared or smudged by the foreground strokes, which can facilitate the exemplar matching process (Section 6.2).

5 Single Stroke Synthesis

RealBrush starts by generating the appearance of a single, isolated stroke. Given an input query spine (a curve-like spline), we estimate a rough 2D shape and u, v parameterization (see Section 4.1). We use the algorithm from Lu et al. [2012] to find an optimal sequence of similar segments from the shape library, by collecting candidate samples via nearest neighbor search and using dynamic programming to find the best fit. Our features are tailored to our task and are similar to Ando and Tsuruno [2010]—we consider turning angle, stroke width, distance to the endpoints, and "appearance" (sampled intensity across the stroke width). Finally, we warp and merge the matched grayscale exemplar segments together (see Figure 7) to determine the lightness, $\mathbf{L}$, of the synthesized stroke. At runtime, given a user-specified query color, we convert it to CIELAB space, ($\mathbf{L}_t$, $\mathbf{a}_t$, $\mathbf{b}_t$). Then the synthesized color $\mathbf{C}_f = (\mathbf{L}_f, \mathbf{a}_f, \mathbf{b}_f)$ of each pixel is: $\mathbf{L}_f = \mathbf{L}$, $\mathbf{a}_f = \alpha\mathbf{a}_t$ and $\mathbf{b}_f = \alpha\mathbf{b}_t$, where $\alpha = (100 - \mathbf{L})/100$. The synthesis outputs are a binary query mask $\mathbf{Q}$ indicating the regions of the canvas that belong to the query, the u, v parameterization, and the colors $\mathbf{C}_f$ that we call foreground color for the rest of the paper.

5.1 Warping and Merging

The matched segments will not exactly match the shape of the query stroke and so must be warped to fit. This is done by sampling the exemplar textures according to the u, v parameterization computed by the MBA library [Hjelle 2001]. Because the segments are already close in shape to the query, warping can usually produce a high quality output.

To merge adjacent segments, alpha blending simply linearly interpolates between the two segments in their overlap region, creating a cross-dissolve. For many types of media with low frequency textures such as watercolor and oil paint, this produces high quality results. Alpha blending is easy to implement and efficient, so we use it for our results unless otherwise indicated. Alpha blending can result in two types of artifacts: ghosting and blurring (Figure 8a). Ghosting occurs when matching cannot find segments with similar appearance at the boundary, and blurring is caused by interpolating mismatched structured texture. In both cases, graph cuts can be used to find an optimal seam between segments, followed by gradient domain compositing to smooth the transition [Kwatra et al. 2003] (Figure 8b). In many cases, graph cuts generate a superior result and is still relatively efficient, so we reserve it as an alternative merging option.

5.2 Texture Synthesis

To better deal with highly textured media (Figure 8), we have prototyped an optional texture synthesis module which runs off-line to produce higher stroke quality. We adapt image melding [Darabi et al. 2012] to the problem of stroke synthesis. As a review, image melding works through multiscale texture synthesis over small square patches. The "melding" component allows for smooth transitions between different textures without losing detail in the transition region. We use this to transition between different brush exemplars. We also use the "texture preserving warps" which improve the texture distortion introduced by warping.

We apply image melding to an initial guess which is the warped image after graph cut. However, naive melding alone produces inferior visual results. Therefore we add constraints to image melding to better guide synthesis. These include index masks, color corrections, and rotation constraints. The constraints are illustrated in Figure 9. The importance of constraints is shown in Figure 10, which gives a comparison with naive image melding.

Index masks. We apply index masks to constrain interior regions in the result to be synthesized from interior regions in the exemplars. We define interior regions to be those that are more than a threshold distance $\tau = 1/4$ from the stroke boundary, where the maximum distance is defined to be unity. The interior regions are shown in Figure 9, regions 1 and 3. Similarly to image melding, we also apply distance constraints, which prevent matched patches from moving too far from the initial correspondence given by warping.

Color corrections. Image melding can introduce undesired fluctuations in color because of its gradient energies. We thus constrain

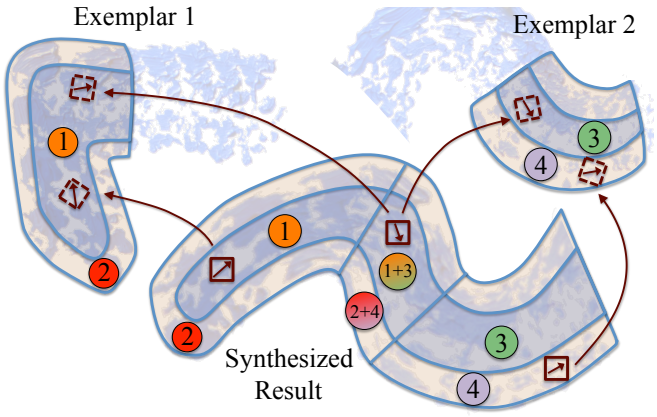


Figure 9: *Explanation of the optional texture synthesis module. Two exemplars (top) are matched and an initial guess is constructed by warping. Texture synthesis improves any texture distortion. Each upright patch (small square) in the result is matched during multi-scale synthesis to an appropriately rotated patch in an exemplar. Rotation constraints orient patches to follow the brush principal angle (indicated by small arrows inside the patches). Distance constraints prevent matched patches from moving too far from the initial correspondence given by warping. Patches are constrained to only come from the same index mask: patches in region 1 match patches in region 1, and so forth. The overlap regions* $1+3$ *and* $2+4$ *match both exemplars and are melded for a smooth transition.*

the color outside the initial guess to be transparent and white. To prevent fluctuations in paint density due to any averaging, at coarse scales after each iteration we use color transfer [Pitié et al. 2007] to match colors to the initial guess. We do this locally by running the color transfer on each of 5x5 sub-windows that are overlapping, and smoothly interpolating the result in the overlapped regions.

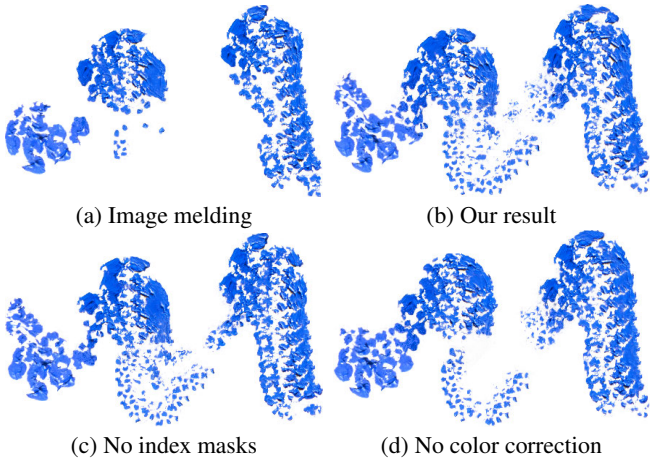
Rotation constraints. We constrain rotations to be similar to the principal brush angle. These angles follow the direction of the brush stroke and are shown as small arrows inside the patches in Figure 9. Patches in the synthesized image are always upright, whereas matched patches in the source exemplars rotate. We allow some deviation from the exact brush principal angle by allowing for matches in an angular window. We use a larger angular window of 30 degrees in the interior region, and a smaller angular window of 10 degrees on the boundary.

6 Stroke Interaction Synthesis

Single stroke synthesis does not consider what is on the canvas. When strokes overlap, independent synthesis will produce artificial results. When real media overlap, their appearances change predictably. RealBrush reproduces two types of these interactions: smearing and smudging.

We observe that smearing and smudging happen mostly within the interaction region (defined in Section 4.2). We factor the interaction effect into three components: color, texture and shape. In most natural media, the impact to the foreground stroke shape is not obvious, so we only handle the change of color and texture. This is a consequence of our natural media factorization—first we perform single stroke synthesis (Section 5), and input the appearance into the smearing and smudging algorithms.

When the user paints a new stroke, our system first detects a set of overlap regions where the canvas beneath already has paint (Section 6.1). Per overlap region, we match to a smudging or smearing exemplar for synthesis (Section 6.2). To warp the exemplar, we vectorize the interaction regions (Section 6.3). To determine the colors



(a) Image melding (b) Our result

(c) No index masks (d) No color correction

Figure 10: *Comparison with image melding. Melding alone can produce results that do not closely follow the brush stroke (a). Our texture synthesis gives an improved result (b). The effect of removing index masks from our algorithm is shown (c), as well as the effect of removing color correction (d). Without our constraints, the texture can deviate from the brush stroke.*

in the interaction region, we apply weighted color integration (Section 6.4). The only difference between smearing and smudging is how we apply the matched exemplar (Section 6.5 and Section 6.6).

6.1 Overlap Regions Extraction

As strokes accumulate on the canvas, a new stroke might overlap many previous strokes creating an exponential number of overlap regions. However, our recursion assumption allows us to treat the canvas as a single, flattened paint raster, which makes the approach efficient and scalable. Specifically, the smearing or smudging behavior of the current time step t is only dependent on the raster canvas image of all strokes at $t-1$, the time of the previous stroke.

At every time step, we update a binary coverage mask $\mathbf{C}_t$ to keep track of whether a pixel on the canvas has paint. After the user paints a new query stroke, the system detects a set of overlap regions $\mathbf{\Omega}_t = \{\mathbf{\Omega}_t^i\}$ by intersecting the query mask $\mathbf{Q}_t$ with the coverage mask: $\mathbf{\Omega}_t = \mathbf{Q}_t \cap \mathbf{C}_t$. Each overlap region $\mathbf{\Omega}_t^i$ is restricted to lie inside the query stroke and forms a connected component in the coverage mask (Figure 13a). We break long, irregularly shaped overlap regions into smaller ones by scanning the connected component along the stroke spine to identify cusps and cutting at the cusp perpendicular to the spine. For example, the first two overlap regions in Figure 13a originate from a single connected component. We also discard small overlap regions, since their influence on the final painting are minimal. The rest of the overlap regions are classified into several categories (Figure 11) for vectorization.

6.2 Exemplar Matching and Feature Vectors

We synthesize the stroke interactions by warping the captured interaction exemplars. To reduce warping artifacts, we search for exemplar strokes that cross at similar angles and have similar thickness. For simplicity and robustness, we design a raster feature vector to capture information of the overlap region and its surroundings. For each overlap region in the query stroke, the feature vector is a two-channel square texture (50 x 50). The square is oriented along the average foreground stroke spine orientation in the overlap region. It contains the entire overlap region with padding so the surroundings can be captured. The first texture channel contains the alpha matte of the background in the overlap

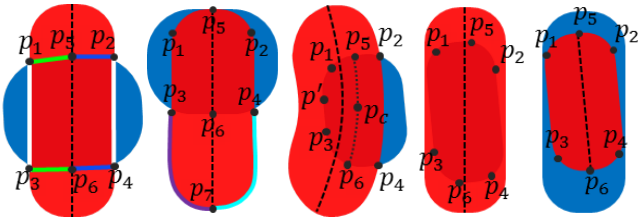


Figure 11: *Stroke overlap situations. (a) - (e) The query stroke (red) might overlap with the background stroke (blue) in several different configurations. The overlap region might (a) run across the query stroke; (b) be at the beginning or the end of query stroke; (c) not separate the query stroke into disjoint regions; (d) be entirely inside the query stroke; (e) be the entire query stroke; The colored line segments in (a) indicate the vectorization of the overlap region. In (b), they indicate the trailing region.*

region. The second texture channel contains a mask of whether strokes are present in the surrounding region. The feature vector distance is the L^2 distance between the squares. When providing the exemplars, the user can optionally input photographs of the background strokes before foreground strokes are applied. After manually aligning the "before" and "after" textures, we sample intensities of the "before" overlap region as the first channel of our feature vector. The use of the "before" image allows the system to search for crossing exemplars not only similar in shape, but also in texture and intensity. Without the "before" image, we reduce the feature vector to one channel. The search is not sensitive to the size of the square or the precision of the "before" and "after" alignment. The amount of distortion even in the case of very different exemplars is hardly noticeable in the context of a whole painting. Figure 12 shows the query stroke feature vector and the most similar overlapping exemplar.

6.3 Interaction Region Vectorization

After identifying the most similar interaction exemplar, we need to warp the matched exemplar to fit the query interaction region exactly for synthesis. Section 6.1 identifies the overlap regions in raster format. We then need to vectorize the regions and identify the trailing regions. We extract the six polylines (green, blue and white lines in Figure 11a) for the overlap regions and two polylines (purple and cyan lines in Figure 11b) for the trailing regions, corresponding to that of the exemplar (Figure 6a).

From the overlap region mask, we extract the overlap contour using OpenCV. We scan the overlap contour to identify six "key points" that divide the contour into six polylines. For overlap regions that cross the query stroke (Figure 11a), we first intersect the stroke spine with the contour to identify $\mathbf{p}_5$ and $\mathbf{p}_6$. We then locate the $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ by tracing from $\mathbf{p}_5$ and $\mathbf{p}_6$ clockwise and counterclockwise along the overlap contour until reaching the query stroke boundary. When the overlap region is at the beginning (or end) of the query stroke (Figure 11b), we identify $\mathbf{p}_5$ to be the first sample on the query spine and apply an offset from $\mathbf{p}_5$ to locate $\mathbf{p}_1$ and $\mathbf{p}_2$ on the query contour. The offset is determined by the average thickness of the query stroke. Then $\mathbf{p}_3$, $\mathbf{p}_4$, $\mathbf{p}_6$ are extracted the same as case (a). There are also cases where the overlap region does not separate the query stroke into disjoint segments (Figure 11c). In this case, we find the overlap contour sample $\mathbf{p}'$ with smallest v coordinate. We offset from $\mathbf{p}'$ to obtain $\mathbf{p}_1$ and $\mathbf{p}_3$ on the overlap contour and use the center of mass $\mathbf{p}_c$ to locate $\mathbf{p}_5$ and $\mathbf{p}_6$ following the stroke spine direction. Then $\mathbf{p}_2$ and $\mathbf{p}_4$ are traced from $\mathbf{p}_5$ and $\mathbf{p}_6$, the same as in case (a). When the entire overlap region is within the query stroke (Figure 11d), none of the points can be located exactly, so we identify points with maximum and minimum u coordinates as $\mathbf{p}_5$ and $\mathbf{p}_6$ and use the offset idea to find the other points. In the final case where the entire query stroke is inside a background stroke (Figure 11e), $\mathbf{p}_5$ and $\mathbf{p}_6$ are the start and end of the stroke spine, and we apply offsets to locate the other points. In real painting scenario, the case in Figure 11c happens frequently resulting in many thin interaction regions. To avoid aliasing and ensure the stableness of the points $\mathbf{p}_i$, we discard overlap regions that are smaller than some threshold. For larger interaction regions, experiments show that the detection of points $\mathbf{p}_i$ is very robust to all possible overlap shapes and angles.

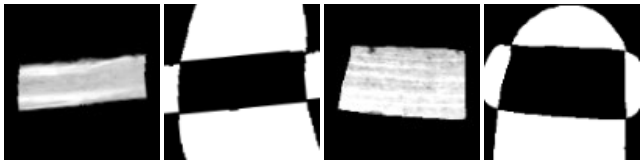


Figure 12: *The overlap region feature vector. The overlap regions of the query and exemplar strokes are characterized by a square two-channel texture. The query feature vector contains the alpha matte of the background in channel 1 (a) and a binary mask of surrounding strokes in channel 2 (b). The best matching exemplar is shown in (c, d).*

To finish vectorizing the interaction region $\mathbf{\Pi}_t^i$, we also detect and vectorize a trailing region $\mathbf{\Psi}_t^i$ that follows the overlap region $\mathbf{\Omega}_t^i$. The trailing region length is determined by the ratio of trailing region length to overlap region length in the exemplar, $L(\mathbf{\Psi}_{query}) = L(\mathbf{\Omega}_{query})L(\mathbf{\Psi}_{exemplar})/L(\mathbf{\Omega}_{exemplar})$, where $L(\mathbf{\Omega}) = (\mathrm{d}u(\mathbf{p}_1,\mathbf{p}_3) + \mathrm{d}u(\mathbf{p}_5,\mathbf{p}_6) + \mathrm{d}u(\mathbf{p}_2,\mathbf{p}_4))/3$ and $L(\mathbf{\Psi}) = (\mathrm{d}u(\mathbf{p}_7,\mathbf{p}_3) + \mathrm{d}u(\mathbf{p}_7,\mathbf{p}_6) + \mathrm{d}u(\mathbf{p}_7,\mathbf{p}_4))/3$. $\mathrm{d}u(\mathbf{p}_1,\mathbf{p}_2)$ defines the absolute difference of u coordinates between two points. We locate the point $\mathbf{p}_7$ by tracing from point $\mathbf{p}_6$ along the stroke spine direction. Then, we trace two polylines from $\mathbf{p}_3$ and $\mathbf{p}_4$ following the stroke spine direction and curve in to reach $\mathbf{p}_7$ (Figure 11b). The trailing region is constrained to lie within the query stroke boundary. When an overlap region is at the end of the stroke, there is no trailing region.

6.4 Interaction Region Color Integration

For smearing or smudging, in each interaction region, the background material colors are picked up by the brush or the finger and smeared into the succeeding foreground stroke regions. The synthesized query stroke color might have contributions from multiple background strokes. For each pixel in the interaction region, we find the background color $\mathbf{C}_b$ by integrating the colors from the pixels on and above (with smaller u coordinates) the current pixel $\mathbf{p}_c$ along the stroke spine direction. The set of the background pixels that contribute to the color of the current pixel is $\mathbf{S} = \{\mathbf{p}_i \mid u(\mathbf{p}_i) < u(\mathbf{p}_c), v(\mathbf{p}_i) = v(\mathbf{p}_c), \mathbf{p}_i \text{ has paint}\}$. Then the integrated background color is calculated as $\mathbf{C}_b = \sum \mathbf{w}_i\mathbf{c}_i / \sum \mathbf{w}_i, \forall \mathbf{p}_i \in \mathbf{S}$, where $\mathbf{c}_i$ is the color of the background pixel $\mathbf{p}_i$ and $\mathbf{w}_i = 1/(\mathrm{d}u(\mathbf{p}_i,\mathbf{p}_c) + \epsilon), \epsilon = 0.06$. We exclude background pixels that have no paint to contribute.

6.5 Smearing Effect Synthesis

As the outcomes of the Single Stroke Rendering pipeline, each pixel of the query stroke has a foreground color $\mathbf{C}_f$. To simulate smearing, at every pixel, some amount of the background color $\mathbf{C}_b$ (defined in Section 6.4) is mixed with the foreground color $\mathbf{C}_f$ (defined in Section 5) to create paint streaking effect. We use the matched smearing operator to estimate the color mixing proportion α at every pixel. Then the synthesized color of each query pixel is a blending of the foreground and background colors,

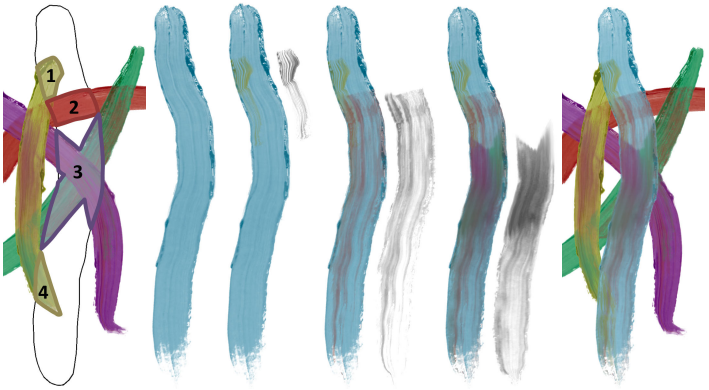


Figure 13: *Smearing synthesis. (a) The query stroke overlaps with the background in four overlap regions; (b) The single stroke synthesis result without smearing; (c) (d) (e) The background colors in the overlap regions are smeared into the subsequent query stroke regions. The warped smearing operators are shown on the right of the query stroke; (f) The four overlap regions are synthesized, alpha composited and rendered in the context of the background.*

$\mathbf{C}_p = \alpha\mathbf{C}_b + (1-\alpha)\mathbf{C}_f$. Specifically, we determine the α for every pixel by warping the smearing operator. We specify control points on the detected polylines (Figure 11a,b) and apply interpolation for all other pixels.

Note that each query stroke can have several overlapping interaction regions. We sort the interaction regions by ascending u coordinates (4 interaction regions in Figure 13a). We synthesize them in order by using color integration, followed by alpha compositing. Let $\mathbf{C}_b^i$ represent the integrated background color for the interaction region i. The smearing result of multiple interaction regions is therefore a recursive alpha compositing of the integrated background color $\mathbf{C}_b^i$ onto the query stroke (Figure 13c-e). $\mathbf{C}^n = \alpha\mathbf{C}_b^n + (1-\alpha)\mathbf{C}^{n-1}$, where $\mathbf{C}^0 = \mathbf{C}_f$ and n is the number of interaction regions. Finally, the updated query stroke colors $\mathbf{C}^n$ are alpha composited onto the canvas (Figure 13f).

6.6 Smudging Effect Synthesis

Smudging proceeds similarly to smearing, except that the foreground query stroke is transparent, so the single stroke rendering pipeline is skipped. Instead, the smudging operator intensities and the integrated background colors $\mathbf{C}_b$ directly determine the smudged colors: the alpha component is taken from the warped exemplar, while the color is $\mathbf{C}_b$. A warped blending attenuation mask is used to decrease alpha at the boundary of the query stroke to avoid boundary artifacts. Finally, the smudged colors are alpha composited onto the canvas. Figure 14 shows a few stroke scribbles demonstrating the smearing and smudging effects. From left to right, the first row is synthesized based on the oil and the lipstick exemplars. The second row uses the pastel and the plasticine exemplars.

7 Evaluation

We conducted three user studies to determine how faithfully our methods reproduce acquired media. Experts can accurately identify our results as computer-generated given sufficient time, so we focus on casual viewers. Talented artists can always work around a system's limitations to make realistic digital paintings by carefully applying many strokes. The resulting images can often fool casual viewers, so evaluating finished paintings is unlikely to yield meaningful data. A stricter standard is to evaluate each algorithm step in isolation. This design controls factors impacting viewers' opinions but outside the scope of the paper, such as stroke trajectories and subjective color preferences.

Figure 14: *Smearing and smudging synthesis results. From left to right, the first row synthesizes the oil and the lipstick media. The second row synthesizes the charcoal and the plasticine media. A few strokes are painted in the background with the smearing effect and then are smudged by a few strokes in the foreground.*

7.1 Study Design

We employ the same basic methodology in all three studies, adapted from that of Lu et al. [2012], which shared a similar overall goal. The idea is to show both real strokes selected from a library, and synthetic strokes based on the same library, asking subjects which ones are real (using a two-alternative forced choice design). If the synthetic strokes effectively match the library, the subject will be unable to differentiate them and will choose randomly, while a less successful synthesis approach may be recognized as fake, and the subject will correctly identify the real stroke more often than 50% of the time. To avoid frustrating the participants, we chose the two media (watercolor and oil) that most people are familiar with and can reliably identify as computer-generated without undue effort. We first describe the study for single stroke synthesis, and after, describe the differences for the other two studies.

Study 1: Single-stroke. Subjects are shown an image (the *guide*) with three photographic examples of real paint strokes in either oil or watercolor. Below it appear two test images, one containing a photograph of a real paint stroke (*ground truth*), and the other showing a (*synthetic*) stroke generated to match the approximate shape of the ground truth stroke. The subject is told that the guide image contains real strokes and asked to pick which of the test images below it is also real (as opposed to a computer-generated fake). When the subject clicks on one of the two test images (believed to be real) they are replaced with a new pair, but the guide image remains unchanged. A progress bar indicates advancement through a series of 26 test pairs until the task is complete.

Of the 26 synthetic images, eight are rendered using the ("*RealBrush*") method of Section 5 (Figure 15c), eight are rendered using a *"naive"* method (Figure 15b), and ten are obvious *"filter"* images (Figure 15d) used to ensure the subject understands the task and is trying. Each test pair is randomly swapped (left-right); a random

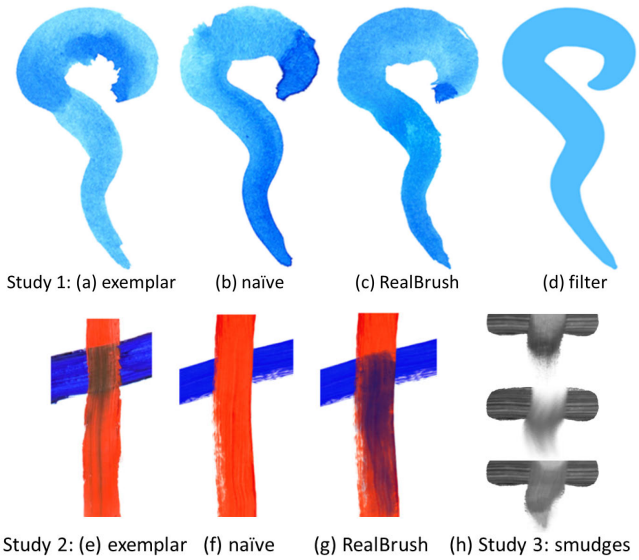


Figure 15: *Study images.*

flip (horizontal, vertical, both, or neither) is applied to both images; and the 26 pairs appear in random order. The eight strokes selected for the RealBrush condition are chosen randomly without repeats to match a pool of 17 (for oil) or 19 (for watercolor) possible ground truth paths. Likewise for the naive and filter conditions. Thus, even though synthetic strokes may only appear once during a task, a particular ground truth path may possibly be seen by the subject as many as three times. However, we find in such cases it is difficult to recognize because of the random flipping and swapping in a long sequence of comparisons (and even if so, it might only weaken our statistical findings).

The strokes in the RealBrush condition follow the paths of their ground truth comparators, and are rendered using a library of 30 strokes, some of which may be used as ground truth in other tests but are prevented from synthesizing their own paths in a hold-one-out scheme. The library strokes are, however, disjoint from the guide image strokes shown to the subject above the test pair. In the naive method, each test stretches a single library example chosen randomly from a set of ten relatively straight library strokes, rather than fitting multiple parts based on shape as in the RealBrush condition. The strokes in the filter condition, whose goal is to be easily recognized, have constant color over the entire stroke, roughly matched to the ground truth. The strokes are all rendered in blue to prevent color preferences from affecting the results.

Study 2: Smearing. The second study attempts to evaluate the effectiveness of using smearing operator to approximate paint streaking behavior in oil media. To avoid the influence of the color choice, we show crossing strokes using blue as background and red as foreground consistently as with all of our exemplars (Figure 15e-g). The ground truth pool contained 10 examples, and the corresponding RealBrush conditions were drawn in our application to roughly match the crossing shape of the ground truth, using 9 possible exemplars in a hold-one-out scheme. The instructions described, "paint strokes crossing over each other." In other respects the study design was as in the first study above.

Study 3: Smudging. In the third study, the guide image showed five smudges of blue paint. However, the test images were shown in black-and-white because the color model used in our system is insufficient to capture the subtlety of this effect well enough to avoid detection when compared with real smudges (Figure 16d,e). Using grayscale allows us to factor out the color influence and only evaluate the effectiveness of the smudging operator. Accordingly the instructions are modified describing "black-and-white photos below." The RealBrush condition smudges are based on a library of 10 exemplars, whereas the naive approach uses the "smudge" tool in Adobe Photoshop. (see Figure 15h.)

study	correct / count (%) naive method	RealBrush
1: single stroke	462 / 736 (63%)	360 / 736 (49%)
2: smearing	703 / 760 (93%)	487 / 760 (64%)
3: smudging	673 / 688 (98%)	413 / 688 (60%)

Table 1: *Results for three user studies. Subjects were asked to identify real vs. synthetic strokes. In one condition the synthesis method was naive, while in the other condition strokes were generated by methods in Sections 5–6. Results are reported as ratios: pairs correctly identified over pairs shown. If the synthesis method effectively matches the medium, subjects will be forced to guess, leading to an expected 50% ratio. On the other hand, less effective approaches will be correctly identified more often, leading to higher ratios. In each study the RealBrush condition is more effective than the naive approach, with statistical significance.*

7.2 Study Results

Our subjects were recruited on Amazon's Mechanical Turk, and restricted to US-only "workers" who were paid $0.20 per task ("HIT"), which they typically completed in a few minutes. In the first study, a single worker could perform as many as five HITs for oil and five for watercolor, while in the second and third studies workers could perform up to five HITs per study. Most workers did just one HIT in any study, and in total 139 unique subjects participated. For each of the three studies 100 HITs were posted, from which, respectively, data from 92, 86 and 95 were retained after omitting HITs from workers who failed to correctly identify the 10 filter images.

Table 1 shows the results of these studies. In each, a Bonferroni corrected, randomized permutation test on the distributions shows that the naive approach and the RealBrush methods perform differently with statistical significance ($p \ll 0.01$). In the case of single strokes the RealBrush method appears to have been indistinguishable from real photographs. While less effective, the smearing and smudging methods still perform well and clearly outperform the naive methods.

8 Results and Discussion

We gave our prototype painting application to artists to experiment with the media we have acquired. Some paintings are in Figure 19. These exhibit a wide range of styles and effects. The flowers and ocean sunset use plasticine for single strokes, smearing, and smudging, resulting in smooth textures. The tiger uses oil paint and little smearing, while the bird uses oil smudging to significant effect. The dancer's simple pencil line work demonstrate the quality of our synthesis for long strokes. The train and landscape mix media, including oil, watercolor, pencil, and lipstick, which underscores the utility of supporting many artistic media. Figure 18 shows that our smudging tool can be used to manipulate photographs for a painterly effect.

The first result of our evaluation is that for common natural media (oil and watercolor), synthesized individual strokes are indistinguishable from real examples to casual viewers. Though not part of the evaluation, other media with similar texture properties such as plasticine, pencil, charcoal, and lipstick, can also be plausibly reproduced. Off-line texture synthesis handles the harder cases such as sponge and glitter strokes. One consideration is that we do not

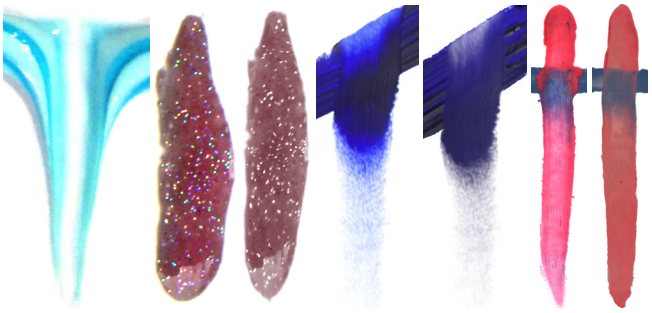


Figure 16: *Synthesis limitations. (a) shows a toothpaste smudging example that will not be successfully reproduced using our synthesis approach. (b-g) shows side by side comparison of an exemplar (left) with the synthesis result (right). (b,c) lip gloss. Only using the lightness of exemplars fails to reproduce the color of the sparkles. (d,e) oil smudging. The textures are faithfully reproduced, but our simple color model fails to mimic hue changes inside the interaction region. (f,g) plasticine smearing. The textures and colors in the interaction region are plausibly reproduced, but the foreground stroke boundary remains untouched leaving an artificially clean look.*

specifically handle lighting artifacts during exemplar acquisition. For example, thick media such as toothpaste (see Figure 16a) may cast shadows, and wet media such as lip gloss (see Figure 16b) may have specular highlights. Since our matching is rotation invariant, synthesized strokes may have inconsistent lighting effects as a result. For our exemplar libraries, we capture the media using overhead lighting to mitigate these artifacts. Another limitation lies in our color replacement model. We only use the lightness of the exemplars to modulate the synthesized strokes, therefore any color differences within the exemplars will be lost. For example, the sparkles in Figure 16b are not well reproduced in the result in Figure 16c. Texture distortion due to warping is reduced by our piecewise matching technique, but can still be observed in some regions, such as Figure 16c and the oil sponge and glitter in Figure 8a.

The second result of our evaluation is that synthesized smears and smudges often look plausible, but are not indistinguishable from real examples. We believe this is largely due to the difference in color mixing, which we leave as future work. For smearing, Figure 15f,g shows that the synthesized color in the interaction region is different from that of the exemplar. For smudging, a hue shift (from dark blue to cyan) can be observed from the exemplar in Figure 16d. Our use of simple alpha blending in the color integration step results in transition from dark blue to gray in Figure 16e. Regardless, in practice this is not a serious limitation as smudges and smears generally occur in complex paintings amid many strokes, where fine details are less noticeable (see Figures 14 and 19). For some media interactions, our factorized model assumptions do not hold. We assume that smearing synthesis does not alter anything outside the foreground stroke's silhouette. However, for some media, the silhouette may be changed slightly, as with plasticine stroke in Figure 16f. Our approach plausibly reproduces the textures and colors within the silhouette, but leaves the foreground stroke shape untouched which gives an artificially clean look (Figure 16g). For smudging, some challenging media such as toothpaste are not handled well by blending the smudging operator with the background strokes. Due to strong textures in toothpaste, the synthesized texture in the interaction region may not match the textures in the background stroke (see Figure 16a).

Figure 17 shows stroke synthesis results from Microsoft Fresh Paint, a commercial oil paint application based on state of the art research [Baxter and Govindaraju 2010; Chu et al. 2010], for comparison with Figure 14. We achieve comparable visual quality, while

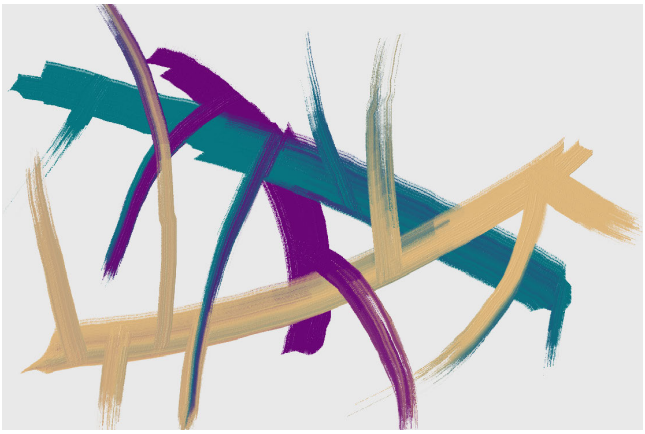


Figure 17: *Test strokes in Microsoft Fresh Paint [Baxter and Govindaraju 2010; Chu et al. 2010], generating oil strokes, smeared colors, and smudging with a dry brush, for comparison with Figure 14.*

our data-driven approach produces richer textures and organic appearance. An important advantage of our approach is our support of a broader range of media. Conversely, our synthesized appearance is harder to predict or control—a small change in the stroke path might lead to different matched exemplars with disparate appearances. However, this is arguably similar to the physical painting process where strokes can have unpredictable appearances, especially for novices. Additionally, since our exemplar strokes encode the experts' techniques, we often found that poorly made query paths result in aesthetically pleasing synthesized strokes.

Currently, we do not support "scribble strokes," where the stroke overlaps itself, due to the use of MBA to compute the u, v parameterization (Section 5.1), which can limit paint mixing on-canvas. This is not a fundamental limitation, as alternative mechanisms for computing the parameterization may not have this problem. Meanwhile, our smudging functionality provides an easy alternative for paint mixing (see Figures 1b and 19e). Another limitation is that synthesis is performed on mouse-up. Both the piecewise matching optimization and the detection of overlap regions require the knowledge of the whole query stroke. Future work might investigate synthesizing mid-stroke, possibly using a sliding window for optimization.

For a 1000x1000 canvas, synthesizing one stroke takes from 0.5 to 1 seconds, depending on the stroke length and library size. Due to the rotation invariant nature of the matching feature vector, roughly 20 strokes with different thickness and curvature profiles are enough to avoid noticeable repetitions of the same exemplar segments. Though increasing the library size diversifies the synthesis appearance, we do not find significant perceptual quality improvement. The matching performance scales sub-linearly with the library size depending on the nearest neighbor search package. For our chosen canvas size, the performance bottom-neck lies in the image manipulations for stroke vectorization, color integration, and exemplar warping. For synthesizing a query stroke, the performance is constant in the number of existing strokes on the canvas, but heavily depends on the canvas resolution. On the contrary, a vector-based approach that does not flatten the canvas might perform badly as the stroke count goes up, which often happen in real painting scenarios. All our pixel manipulation is implemented in unoptimized C++ on a single core, so could be improved significantly. The texture synthesis takes minutes per stroke, but large portions are implemented in MATLAB and may be optimized.

Conclusion. We present RealBrush, a fully data-driven system for reproducing high fidelity natural media effects in real-time. We have shown that RealBrush is usable by artists and creates re-

Figure 18: *An additional benefit of our recursion assumption (Section 3) is manipulating photographs. Here, lipstick smudges were used for a painterly effect. The original photo is inset.*

sults that casual users cannot distinguish from photographs. Furthermore, we propose a factorized representation of natural media painting with explicit assumptions about physical properties that can motivate future research, and we make our data available to aid that goal.

Acknowledgements

Special thanks to the contributing artists Daichi Ito, Chen Lifshitz and Daniela Steinsapir. This work was supported by Adobe.

References

ANDO, R., AND TSURUNO, R. 2010. Segmental brush synthesis with stroke images. In *Proceedings of Eurographics – Short papers*, 89–92.

BAXTER, W., AND GOVINDARAJU, N. 2010. Simple data-driven modeling of brushes. In *Proceedings of I3D*, 135–142.

BAXTER, W. V., WENDT, J., AND LIN, M. C. 2004. IMPaSTo: A realistic, interactive model for paint. In *Proceedings of NPAR*, 45–56.

CHU, N., AND TAI, C.-L. 2005. MoXi: Real-time ink dispersion in absorbent paper. In *Proceedings of SIGGRAPH*, 504–511.

CHU, N., BAXTER, W., WEI, L.-Y., AND GOVINDARAJU, N. 2010. Detail-preserving paint modeling for 3d brushes. In *Proceedings of NPAR*, 27–34.

DARABI, S., SHECHTMAN, E., BARNES, C., GOLDMAN, D., AND SEN, P. 2012. Image melding: combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics 31*, 4, 82:1–82:10.

DIVERDI, S., KRISHNASWAMY, A., AND HADAP, S. 2010. Industrial-strength painting with a virtual bristle brush. In *Proceedings of Virtual Reality Science and Technology*, 119–126.

DIVERDI, S., KRISHNASWAMY, A., MĚCH, R., AND ITO, D. 2012. A lightweight, procedural, vector watercolor painting engine. In *Proceedings of I3D*, 63–70.

HJELLE, Ø. 2001. Approximation of scattered data with multilevel b-splines. Tech. rep., SINTEF.

HSU, S. C., AND LEE, I. H. H. 1994. Drawing and animation using skeletal strokes. In *Proceedings of SIGGRAPH*, 109–118.

KIM, M., AND SHIN, H. J. 2010. An example-based approach to synthesize artistic strokes using graphs. *Computer Graphics Forum 29*, 7, 2145–2152.

KWATRA, V., SCHODL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3, 277–286.

LU, J., YU, F., FINKELSTEIN, A., AND DIVERDI, S. 2012. Helpinghand: Example-based stroke stylization. In *Proceedings of SIGGRAPH*, 46:1–46:10.

PITIÉ, F., KOKARAM, A., AND DAHYOT, R. 2007. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding 107*, 1, 123–137.

RITTER, L., LI, W., CURLESS, B., AGRAWALA, M., AND SALESIN, D. 2006. Painting with texture. In *Proceedings of the Eurographics conference on Rendering Techniques*, 371–376.

SCHRETTER, C. 2005. A brush tool for interactive texture synthesis. *ICGST International Journal on Graphics, Vision and Image Processing 6*, 5, 55–60.

VAN HAEVRE, W., VAN LAERHOVEN, T., DI FIORE, F., AND VAN REETH, F. 2007. From dust till drawn: A real-time bidirectional pastel simulation. *The Visual Computer*, 925–934.

WANG, B., WANG, W., YANG, H., AND SUN, J. 2004. Efficient example-based painting and synthesis of 2D directional texture. *IEEE Trans. Visualization and Computer Graphics*, 3, 266–277.

XIE, N., LAGA, H., SAITO, S., AND NAKAJIMA, M. 2011. Contour-driven sumi-e rendering of real photos. *Computers & Graphics 35*, 1, 122–134.

XIE, N., HACHIYA, H., AND SUGIYAMA, M. 2012. Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. In *Proceedings of the International Conference on Machine Learning*, 153–160.

XU, M., AND DONG, J. 2006. Generating new styles of chinese stroke based on statistical model. In *Proc. International Multiconference on Computer Science and I.T.*, 215–222.

XU, S., TANG, M., LAU, F. C. M., AND PAN, Y. 2004. Virtual hairy brush for painterly rendering. *Graphical Models 66*, 5.

XU, S., XU, Y., KANG, S. B., SALESIN, D. H., PAN, Y., AND SHUM, H.-Y. 2006. Animating chinese paintings through stroke-based decomposition. *ACM Transactions on Graphics 25*, 2, 239–267.

XU, S., TAN, H., JIAO, X., LAU, F., AND PAN, Y. 2007. A generic pigment model for digital painting. *Computer Graphics Forum 26*, 609–618.

YAN, C.-R., CHI, M.-T., LEE, T.-Y., AND LIN, W.-C. 2008. Stylized rendering using samples of a painted image. *IEEE Trans. Visualization and Computer Graphics 14*, 2, 468–480.

ZENG, K., ZHAO, M., XIONG, C., AND ZHU, S.-C. 2009. From image parsing to painterly rendering. *ACM Transactions on Graphics 29*, 1.

ZHOU, S., LASRAM, A., AND LEFEBVRE, S. 2013. By-example synthesis of curvilinear structured patterns. *Computer Graphics Forum 32*, 2.

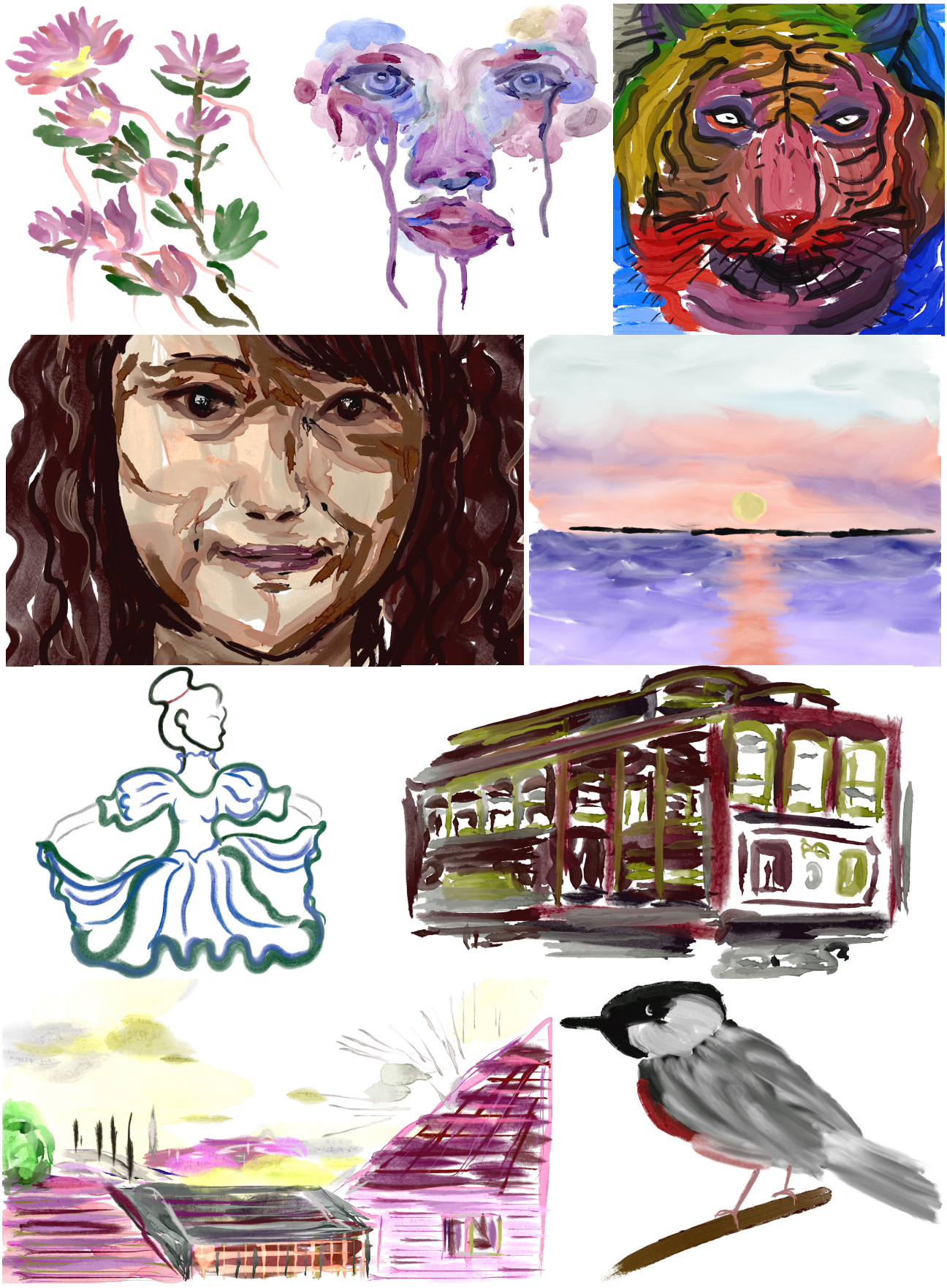


Figure 19: *Example artwork by contributing artists. Used with permission. Details in Section 8.*